Particle-Particle, Particle-Scaling function (P3S) algorithm for electrostatic problems in free boundary conditions.

Alexey Neelov, S. Alireza Ghasemi, Stefan Goedecker
Institute of Physics, University of Basel, Klingelbergstrasse 82, CH-4056 Basel, Switzerland

Abstract

An algorithm for fast calculation of the Coulombic forces and energies of point particles with free boundary conditions is proposed. Its calculation time scales as $N \log N$ for N particles. This novel method has lower crossover point with the full $O(N^2)$ direct summation than the Fast Multipole Method. The forces obtained by our algorithm are analytical derivatives of the energy which guarantees energy conservation during a molecular dynamics simulation. Our algorithm is very simple. An MPI parallelised version of the code can be downloaded under the GNU General Public License from the website of our group.

1 Introduction

The computation of the electrostatic or gravitational interaction of a large number N of point particles is a central problem in many fields of physics, such as molecular dynamics [1] and astrophysics [2].

If the charge distribution is nonperiodic, one frequently employs the fast multipole method (FMM) [3]- [8]. Its computation time scales as $O(N)$. Its point of the crossover with full direct calculation depends on the accuracy and is estimated to be $10^3 - 10^4$ in [5] or $10^4 - 10^5$ in [9]. The Barnes and Hut hierarchical tree method [10] is advocated for the use with low accuracy ($0.1 - 1\%$) in [11].

The cutoff methods were recently revived [12], [13]. They also scale as $O(N)$, but because of the immense prefactor they are only used for special systems.

If the charge distribution is periodic, one can use the Ewald method [14]-[16], that scales as $O(N^{3/2})$. It is good for up to $10^3 - 10^4$ particles and can achieve high accuracy; it is clearly faster than full direct summation of the periodic particle images.

The Ewald method was further improved by the use of Fast Fourier Transform; that led to the particle-mesh schemes [17]- [28]. These algorithms scale as $O(N \log N)$. The same scaling was obtained by combining the Ewald method with the FMM for the calculation of the short range forces [29] and, alternatively, by using nonuniform FFT in the Ewald method [30]. The particle-mesh schemes are suitable [11] for relatively high RMS force error of about 10^{-4} and $N = 10^4 - 10^5$.

The FMM method can be applied in the periodic case too [6], [22], [31] - [33] especially if one needs high precision. Due to its better scaling, it becomes preferable in the limit of large N. However for not so big N the particle-mesh schemes are faster. The exact location of the crossover point depends both on the system studied and the computer used [11], [33]. The usual estimate for the crossover is around $N = 10^5 - 10^6$.

It is mentioned in [1], [34] that in the FMM the forces are not equal to the negative analytical gradients of the energy. Therefore if one uses the FMM for an MD simulation, the total energy will change during the simulation, unless very high precision is used - see e.g. Fig.1 of [32]. Thus it is impossible to do simulations in the microcanonical ensemble.

In the particle-mesh schemes it is possible to have conservation of either energy or momentum, but not the two together [20], [21]. The particle-mesh schemes are usually also easier to code, compared to the FMM.

Much attention has been given to the parallelization of the particle-mesh [22], [25], [35]- [37], and especially FMM [38]- [41] algorithms. In the case of FMM, the parallelization is more difficult technically, but more efficient. Recent papers [42], [43] cite speedups of over 100 for 10^5 particles for a state of the art parallel implementation of FMM. In contrast, the parallel speedups of the simple particle-mesh methods of [35], [37] do not exceed 10 for 10^5 particles; there is probably room for improvement.

Many systems ranging from those in the electronic structure calculations of molecules to those of astrophysics require the use of free boundary conditions (BC). Therefore the imposition of artificial periodicity that is needed for the Ewald-type methods can lead to complications [44], [45]. It would be useful to have a particle-mesh-like method that could handle free BC.

Such algorithms were set forth in [46], [47], [48]. The Hockney method [17] was intended for applications in plasma physics where high precision is not required. Martyna and Tuckerman [47] used the SPME method of Essman et al. [20], along with a special scheme to handle free BC. That scheme was also used for the solution of Poisson equation for smooth charge distributions occurring in electronic structure calculations. However, the method of [47] has the disadvantage that the accuracy decreases if one approaches the border of the cell. Thus one needs to take bigger cells that leads to an increase of the CPU time needed.

Another line of the particle-mesh like algorithms stems from the Fast Fourier Poisson method [19], [25]. In that case one solves the Poisson equation for the auxiliary charge distribution that is made up from Gaussians centered at the original particle positions. The resulting charge density is then projected onto a grid and the thus discretized Poisson equation is solved via an FFT. The forces are obtained by analytical differentiation of the energy expression; this solves the aforementioned problem of energy conservation. This method is believed to be accurate but slow [1] compared to the standard particle-mesh schemes such as P3M [17] or SPME [20].

It was proposed in [49] to use multigrid [50] for the solution of the Poisson equation, instead of the Fast Fourier Transform. This technique was further developed in [25], [27], [28]. The CPU time in the multigrid algorithm scales as $O(N)$; however due to large prefactor it becomes preferable to the FFT-based methods only for a very large N and/or on a massively parallel computer.

The above Fast Fourier-Poisson methods are only applicable with the periodic BC. However, in the new method of Sutmann and Steffen [48] the free BC were accounted for by calculating the potential at the boundaries with the FMM algorithm. This algorithm may be a competitor to the FMM algorithm in the future. Another particle-mesh-like algorithm that uses an iterative solver instead of FFT is described in [26].

In a recent paper [51] we proposed a new algorithm for the solution of the Poisson equation for smooth charge distributions with free BC. It is also MPI-parallelised.

In the present paper we propose a combination of the Poisson solver with free BC [51] with the Fast Fourier Poisson method [19]. We call it particle-particle particle-scaling function (P3S) algorithm to emphasize the relation to the particle-mesh schemes. Our method can achieve high accuracy and speed and compete with the FMM schemes in the range of particle numbers $N = 10^3 - 10^5$ that is important in many applications. The approximate forces resulting from our method are exact (negative) analytic gradients of the approximate energy, which allows the conservation of energy during MD runs.

To calculate the short-range forces, we make a linked list [15]. Then, following [52], [53], we rearrange the particles to optimize the cache performance.

An MPI parallelised version of our code can be downloaded under the GNU General Public License from the website of our group,
http://pages.unibas.ch/comphys/comphys/SOFTWARE/index.html.

The paper is organized as follows. Section 2 describes the Ewald construction in free BC. In Section 3 we briefly review the Poisson solver of [51] for the calculation of the long range Ewald forces and energies for free BC . In Sections 4 and 5 we discuss the cutoffs for the short and long range forces, respectively. In Section 6 we describe the linked cell list for the acceleration of the short range force calculation. Section 7 contains the final formulas for the interparticle forces. In Section 8 we give the results of our code for systems of 1000-20000 particles and give the graphs of the optimal parameter values for a given accuracy. Section 9 describes the results of an MD simulation of a 1000-particle NaCl crystal that demonstrates the energy conservation property of our algorithm. Finally, in Section 10 we describe an efficient parallelization of our algorithm and present the parallel speedup results on a CRAY XT3.

2 The Ewald construction in free boundary conditions [22].

The total electrostatic energy of N point charges in free BC is given by

$$U = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{Q_iQ_j}{r_{ij}}, \qquad r_{ij} \equiv |\mathbf{r}_i - \mathbf{r}_j|.$$

Adding and subtracting the term corresponding to the electrostatic energy of smooth point charges with density $\rho_i(\mathbf{r})$, we get:

$$U = \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\left[\frac{Q_iQ_j}{r_{ij}} - \int\frac{\rho_i(\mathbf{r})\rho_j(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}'\right] +$$
$$+\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\int\frac{\rho_i(\mathbf{r})\rho_j(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}' - \frac{1}{2}\sum_{i=1}^{N}\int\frac{\rho_i(\mathbf{r})\rho_i(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}'.$$

The Ewald choice for the screening charge distribution is

$$\rho_i(\mathbf{r}) = \gamma(\mathbf{r}-\mathbf{r}_i), \qquad \gamma(\mathbf{r}) \equiv Q_i(G^2/\pi)^{3/2}\exp[-G^2\mathbf{r}^2]. \tag{1}$$

We have also experimented with the variant $\gamma(\mathbf{r}) = A(r_{\text{cut}}^2 - r^2)^m$; $m = 4, 8, 16$, where the factor A normalizes the charge to one. However, the CPU time spent by the program with this screening distribution was slightly longer than with the Gaussian for the same accuracy.

The sum of the screening charge distributions is

$$\rho(\mathbf{r}) = \sum_{i=1}^{N} \rho_i(\mathbf{r}). \tag{2}$$

Then, (1) assumes the form

$$\begin{aligned} U &= E_{\text{short}} + E_{\text{long}} - E_{\text{self}}, && (3)\\ E_{\text{short}} &= \frac{1}{2}\sum_{i=1}^{N}\sum_{j\neq i}^{N}\frac{Q_iQ_j}{r_{ij}}\operatorname{erfc}\left(\frac{Gr_{ij}}{\sqrt{2}}\right), && (4)\\ E_{\text{long}} &= \frac{1}{2}\int\frac{\rho(\mathbf{r})\rho(\mathbf{r}')}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\mathbf{r}\mathrm{d}\mathbf{r}', \qquad E_{\text{self}} = \frac{G}{\sqrt{2\pi}}\sum_{i=1}^{N}Q_i^2. && (5) \end{aligned}$$

3 Calculation of the long range energy using the interpolating scaling functions.

The long range term E_{long} in (5) is nothing but the electrostatic energy of the smooth charge distribution $\rho(\mathbf{r})$. Therefore it can be evaluated by a suitable Poisson solver. We make use of the one from Ref. [51] with free BC. It amounts to expanding the screening charge density (2) in a real space basis defined on the grid with spacing h:

$$\begin{aligned} \rho(\mathbf{r}) &\approx \widetilde{\rho}(\mathbf{r}) = \sum_{\mathbf{i}} \rho_{\mathbf{i}}\Phi_{\mathbf{i}}(\mathbf{r}), \qquad \mathbf{i} \equiv (i_1, i_2, i_3), && (6)\\ \Phi_{\mathbf{i}}(\mathbf{r}) &= \Phi(x/h - i_1)\Phi(y/h - i_2)\Phi(z/h - i_3). \end{aligned}$$

It was suggested in [51] to take the interpolating scaling functions [54], [55] of high order (up to 100) as the basis functions $\phi(x)$. The scaling functions of the order N interpolate the polynomials of the order N exactly and are reasonably localized. Therefore they can interpolate a Gaussian very well.

On the other hand, since the scaling functions are cardinal, we obtain for the coefficient in (6):

$$\rho_{\mathbf{i}} = \rho(\mathbf{i}h). \tag{7}$$

The action of calculating this screening distribution on the grid is called the charge assignment in the standard P3M schemes.

Consider the potential that arises from the approximate charge distribution $\widetilde{\rho}(\mathbf{r})$ in (6):

$$\phi(\mathbf{r}) = \int \frac{\widetilde{\rho}(\mathbf{r})}{|\mathbf{r}-\mathbf{r}'|}\mathrm{d}\mathbf{r}'. \tag{8}$$

At a grid point $\mathbf{j}$, this potential has the form

$$\phi(\mathbf{j}h) \equiv \phi_{\mathbf{j}} = h^2 \sum_{\mathbf{i}} K_{\mathbf{i}-\mathbf{j}} \rho_{\mathbf{i}},$$

where the kernel

$$K_{\mathbf{i}} \equiv \int \frac{\Phi_{\mathbf{i}}(\mathbf{r}) \mathrm{d}\mathbf{r}}{|\mathbf{r}|}$$

is computed in [51].

From this moment on we can use the grid sum approximation to the long range energy:

$$E_{\text{long}} \approx \frac{h^3}{2} \sum_{\mathbf{i}} \rho_{\mathbf{i}} \phi_{\mathbf{i}} = \frac{h^5}{2} \sum_{\mathbf{ij}} \rho_{\mathbf{i}} \rho_{\mathbf{j}} K_{\mathbf{i}-\mathbf{j}}. \tag{9}$$

The latter sum is a convolution that can be calculated via FFT techniques [51]. The kernel $K_{\mathbf{i}}$ is calculated only once at the beginning of a calculation and does not change. Thus the use of high order of interpolation does not lead to a significant increase of calculation time [51].

4 The long-range part cutoff.

The density array in (9) is defined via (7):

$$\rho_{\mathbf{i}} = \rho(\mathbf{i}h) = \sum_{j=1}^{N} \gamma(\mathbf{i}h - \mathbf{r}_j).$$

However, the Gaussian (1) is a quickly decaying function. Therefore, one can make the summation in the above equation only for the charges within the distance x_{cut} from the grid point:

$$\rho_{\mathbf{i}} \approx \sum_{|\mathbf{i}h - \mathbf{r}_j| < x_{\text{cut}}} \gamma(\mathbf{i}h - \mathbf{r}_j). \tag{10}$$

The cutoff distance is chosen so that the function (1) has a small value at the radius x_{cut}.

Actually, the sum in (10) is calculated slightly differently in our program. The charge position $\mathbf{r}_j$ is replaced by the position $h\mathbf{i}_j$ of the grid point that is nearest to it:

$$\rho_{\mathbf{i}} \approx \sum_{(\mathbf{i}-\mathbf{i}_j)^2 < x_{\text{cut}}^2/h^2} \gamma\big(h(\mathbf{i} - \mathbf{i}_j)\big). \tag{11}$$

The error entailed by this is negligible. Then we precalculate and store the square roots of integers in order to determine to which gridpoints a given charge contributes. This is faster than calculating square roots of real numbers and truncating to integers in (10).

As a result the calculation time of the charge spreading and long-range force interpolation is reduced allmost by a factor of two compared to a simple summation over a rectangular domain. Note that this is only possible because in the FFP algorithm the charge density assigned to the grid is made of Gaussians. In other particle-mesh schemes B-splines are used instead, so our method would be inapplicable there.

Making the cutoff approximation in (9) gives us, finally:

$$E_{\text{long}} \approx \frac{h^5}{2} \sum_{\mathbf{ij}} \rho_{\mathbf{i}} \rho_{\mathbf{j}} K_{\mathbf{i}-\mathbf{j}}. \tag{12}$$

The long-range energy of our algorithm is given only by the formulas (11),(12). Our algorithm is thus as simple in formulation as the Fast Fourier Poisson method [19]. On the other hand, the properties of interpolating scaling functions allow us to combine fast Gaussian charge assignment and very high order interpolation. This is to be contrasted e.g. to the use of Lagrange interpolation in the PME method [18] where the values of the interpolating functions have to be actually computed at the particle positions. A good account of this method is given e.g. in Ref. [57]. If one uses Lagrange interpolation of the order L, then the length of the Lagrange interpolation filter is $L+1$ and the number of grid points to which a single charge is assigned is thus $(L+1)^3$. For large L this means that a lot of CPU time will be spent on the charge assignment and force back interpolation. To avoid this, in the PME one uses only low order interpolation.

However, we would like to repeat that our method treats systems with free BC whereas the standard particle-mesh ones are primarily for the periodic systems.

To illustrate the accuracy of our approximation, we calculate the self-energy of a single unit Gaussian (1) by the formula (12). The Gaussian is initially at a grid point and then gradually shifted by two grid constants to see the dependence of the error on its position relative to the grid. We have $x_{\text{cut}} = 4.5h$ and $h = 0.7$; these are typical parameter values that we use in other tests .

The graph of the resulting energy error vs. the shift distance is given at Fig. 1. We see that the error is rather smooth; there is no visible jumps coming from the cutoff approximation. The energy oscillation is due to the grid discretization; its amplitude goes to zero in the limit $h \to 0$

5 The short-range part cutoff.

The short-range pairwise potential in (4) is a rapidly decaying function. Therefore, it is standard to introduce the cutoff radius r_{cut} beyond which the short-range potential is approximated by zero:

$$E_{\text{short}} = \frac{1}{2} \sum_{i,j:\, r_{ij} < r_{\text{cut}}} \frac{Q_i Q_j}{r_{ij}} \operatorname{erfc}\left(\frac{G r_{ij}}{\sqrt{2}}\right). \tag{13}$$

To summarize, the Coulombic energy of N charges in our approximation has the form (3) with the approximate long and short contributions given by (12),(13), correspondingly.

The evaluation of the error function in (13) is computationally inefficient, compared to algebraic operations. Therefore, we replace the term that corresponds to error function by a spline approximation in the interval $(0, r_{\text{cut}})$.

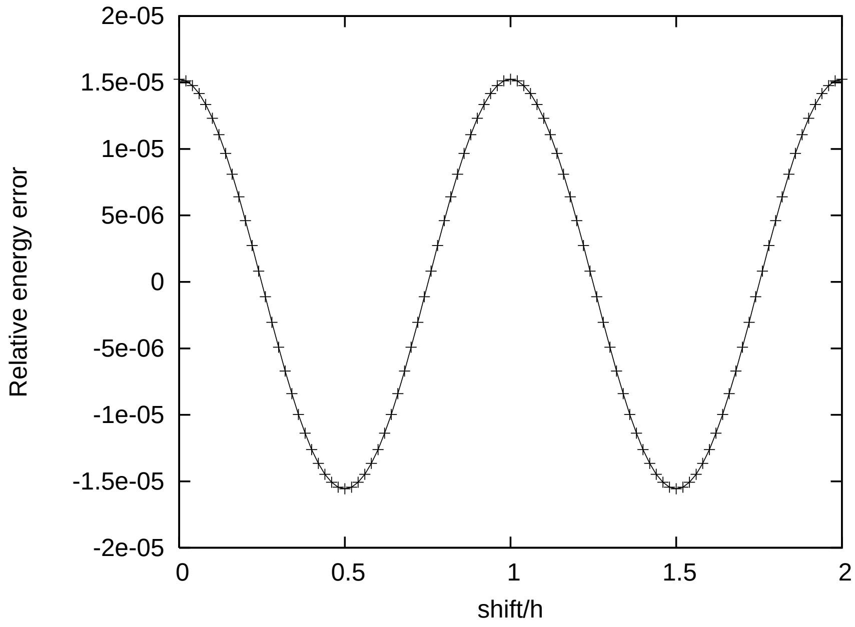

Figure 1: Relative error in the self-interaction energy of a single Gaussian .

The function $\mathrm{erfc}\,(Gr/\sqrt{2})/r$ has a singularity at 0. Therefore we rewrite it as

$$\mathrm{erfc}\,(Gr/\sqrt{2})/r = 1/r - \mathrm{erf}\,(Gr/\sqrt{2})/r, \tag{14}$$

and approximate with a spline only the second term which is regular. This approximation holds for $r < r_{\mathrm{cut}}$. For $r > r_{\mathrm{cut}}$, the whole potential $\mathrm{erfc}\,(Gr/\sqrt{2})/r$ is set to 0.

6 Calculating the forces.

The force acting on the i-th particle is given by the gradient of (3):

$$\mathbf{F}_i = \frac{\partial U}{\partial \mathbf{r}_i} = \mathbf{F}_i^{\mathrm{short}} + \mathbf{F}_i^{\mathrm{long}};$$
$$\mathbf{F}_i^{\mathrm{short}} = Q_i \sum_{j:r_{ij}<r_{\mathrm{cut}}}{}' Q_j \left(\frac{G}{r_{ij}^2}\sqrt{\frac{2}{\pi}} \exp\left[-\frac{G^2 r_{ij}^2}{2}\right] + \frac{1}{r_{ij}^3}\mathrm{erfc}\left[\frac{G r_{ij}}{\sqrt{2}}\right]\right)\mathbf{r}_{ij}, \tag{15}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$; the prime at the sum indicates that it is performed over $j \neq i$. The short range forces (15) are equal to the negative analytical gradients of the short range energy (4). They are also calculated via a spline approximation. Actually, the spline that we use for (14) is obtained by integration of the spline for (15).

Following [25], we also get the long range forces as the negative analytical gradients of (12):

$$\mathbf{F}_i^{\text{long}} \approx h^5 \sum_{\mathbf{lj}} \frac{\partial \rho_{\mathbf{l}}}{\partial \mathbf{r}_i} \rho_{\mathbf{j}} K_{\mathbf{l}-\mathbf{j}} \equiv h^5 \sum_{\mathbf{lj}} \mathbf{q}(\mathbf{l}h - \mathbf{r}_i) \rho_{\mathbf{j}} K_{\mathbf{l}-\mathbf{j}}, \tag{16}$$

where

$$\mathbf{q}(\mathbf{r}) = \frac{\partial \gamma(\mathbf{r})}{\partial \mathbf{r}} = -2\mathbf{r}G^2\gamma(\mathbf{r})$$

is the vector of derivative Gaussians. The calculation of the long range forces (16) is called the force back interpolation.

Taking into account the cutoff (11), one can rewrite the long range force as

$$\mathbf{F}_i^{\text{long}} = h^5 \sum_{\mathbf{j},\mathbf{l}:(\mathbf{l}-\mathbf{i}_j)^2 < x_{\text{cut}}^2/h^2} \mathbf{q}(\mathbf{l}h - \mathbf{r}_i) \rho_{\mathbf{j}} K_{\mathbf{l}-\mathbf{j}}.$$

7 Linked cell list.

In order to avoid the scanning of all particle pairs in (13) we make a linked subcell list [15]. The subcell size is smaller or equal to $r_{\text{cut}}/\text{MMAX}$ where MMAX is a positive integer ranging, in practice, from 1 to 3. The standard linked cell list algorithm corresponds to MMAX $= 1$; higher values of MMAX correspond to smaller cells such as those of [52], [53]. To account for the free BC we add MMAX layers of empty cells outside the original cell. The sum of forces for a given particle is then done over all particles in subcells which are within r_{cut} of the subcell where the original particle is located.

The sum over subcells is done along the stripes in the direction x. In particular, following [52], [53], we rearrange the particles so that the particles in the same subcell have consecutive numbers and the subcells are sorted in the x direction. The actual sorting can be avoided because we already have the linked subcell list. In more detail,

- The first and last particle numbers in each subcell are stored in separate arrays HEAD and TAIL.
- If the cell is empty, its HEAD value is that of the neighboring cell in the direction of increasing x coordinate. For the cells outside the simulation box (to the right of it), HEAD $= N + 1$ by default.
- The TAIL value of an empty cell is that of its neighbor in the direction of decreasing x. For the outside cells (to the left of the simulation box), TAIL $= 0$.
- The loop over all particles in the cell stripe:

  $$(i_1, i_2, i_3), (i_1 + 1, i_2, i_3), \ldots, (j_1, i_2, i_3)$$

  is done as a loop the over particle numbers from HEAD(i_1, i_2, i_3) to TAIL(j_1, i_2, i_3). In this way it does not matter whether empty cells occur.

The reordering of the particles has the additional advantage that the cache performance of the long range part is optimized. The reason is that the particles that have close indices are also close physically: $|\mathbf{r}_i - \mathbf{r}_j|$ is small when $|i-j|$ is small.

The loop in the charge assignment and force back interpolation goes over the particle numbers. The particles in the same subcell have overlapping screening charge densities, so the elements of the density/potential array are reused again and again, as the loop index goes over the particles in the same subcell.

This makes the program significantly faster compared to the case without reordering, especially if the particles are distributed randomly.

8 The results of the serial code.

We chose two test systems:

- N particles in the unit cube with random coordinates and charges equal to ± 1, the total charge being zero (henceforth called the random system)
- N particles with charges ± 1 forming a rock salt crystal, filling the unit cube with $M \approx N^{1/3}$ nodes at each side. The lattice constant is then $d = (M-1)^{-1}$. Each particle is then shifted away from its node by a vector with random coordinates in the range $(-d/3, d/3)$. This system will be called the crystal one.

In both cases, the particle number $N = 1000 * 10^{j/3}$; $j = 0 \dots 4$. In other words, we consider the following values of N: $1000, 2154, 4642, 10000, 21544$.

We always use the scaling functions of the order 100. We found that using lower orders leads to a decline in accuracy, while the order of the scaling functions only affects the time of the calculation of the kernel which is done only once for a simulation.

The values of x_{cut}, r_{cut}, G and h are free parameters. As explained e.g. in [57], such parameters should be chosen such that

- The accuracy is fixed to some value.
- The CPU time spent is minimal for the given accuracy.

As the measure of the accuracy we use the mean square force error:

$$\delta F \equiv \sqrt{\frac{\sum_{i=1}^{N} (\mathbf{F}_i - \mathbf{F}_i^{\text{direct}})^2}{\sum_{i=1}^{N} (\mathbf{F}_i^{\text{direct}})^2}},$$

where

$$\mathbf{F}_i^{\text{direct}} = Q_i \sum_{j \neq i} Q_j \frac{\mathbf{r}_i - \mathbf{r}_j}{|\mathbf{r}_i - \mathbf{r}_j|}$$

are the forces obtained by the full direct calculation.

Unfortunately we cannot give the *a priori* error estimates of the accuracy. The error in the short-range forces is the same as that for the Ewald method, and is estimated in [56], [57]. However, our calculation of the long range forces involves the use of a finite basis and the approximation in Eq. (9), the accuracy of which is difficult to estimate.

Therefore, for our two test systems we made tests with all reasonable values of the parameters. The most important parameter is G, the width of the Gaussian in (1). We also introduced the following dimensionless quantities: the dimensionless Gaussian cutoff Gx_{cut}; the cutoff ratio $r_{\text{cut}}/x_{\text{cut}}$, the dimensionless grid constant Gh and

$$N_{\text{short}} \equiv \frac{4/3\pi r_{\text{cut}}^3}{V_{\text{box}}} N.$$

The latter is the average number of particles inside a sphere of radius r_{cut}. It turns out that the relative MSQ force error depends stronger on these dimensionless quantities and G than on the number of particles in the system.

We checked 7-10 possible values for each parameter. From the resulting pool of results we selected the so called Pareto frontier. A point is on the Pareto frontier if there is no other point which has both smaller CPU time and smaller error.

The Pareto frontiers for the values of $N = 1000 \times 10^{j/3}$; $j = 0 \ldots 4$ for the random system and the crystal systems are given at the Fig.2 and Fig.3. The tests where done on an 1.8GHz AMD Opteron 244.

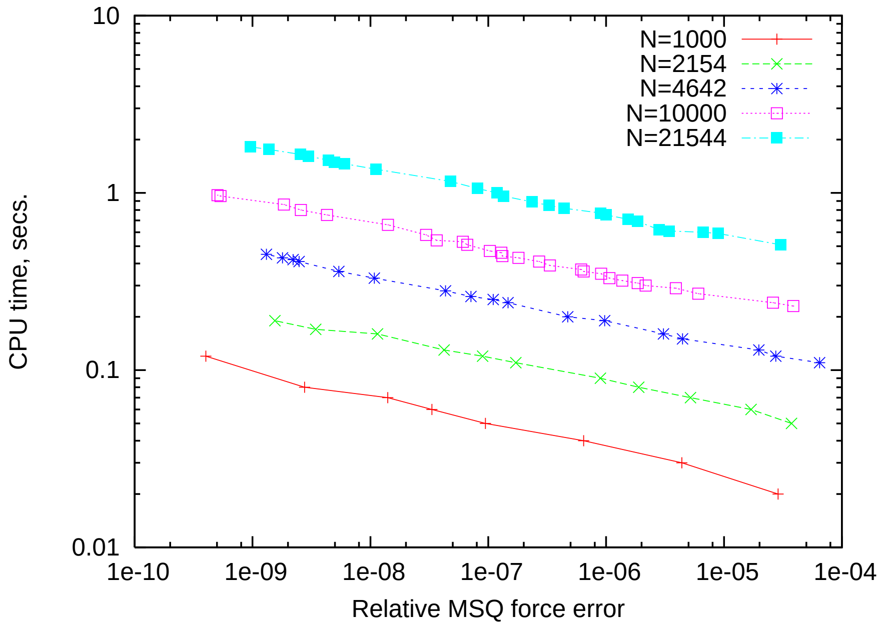

Figure 2: The Pareto frontiers for the random system on Opteron 244.

For the purpose of illustration we present here the optimal parameter values for the crystal system. The values for the random one do not differ significantly.

From Figs. 4,5 we see that the optimal values of Gx_{cut} and Gh are determined by the accuracy level. In contrast, the optimal value of N_{short} depends on the number of particles too. Therefore in an actual calculation it must be adjusted by the trial and error method to give optimal accuracy and CPU time.

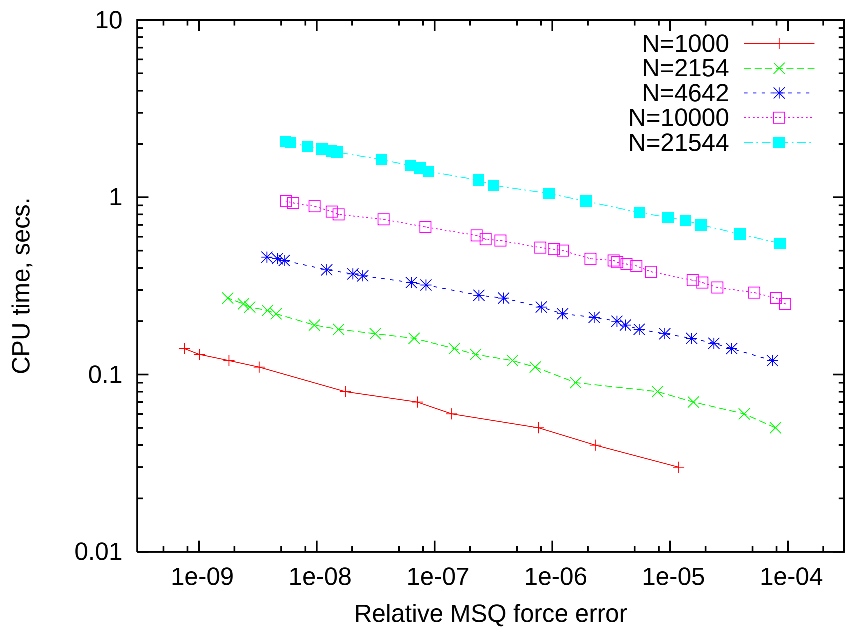

Figure 3: The Pareto frontiers for the crystal system.

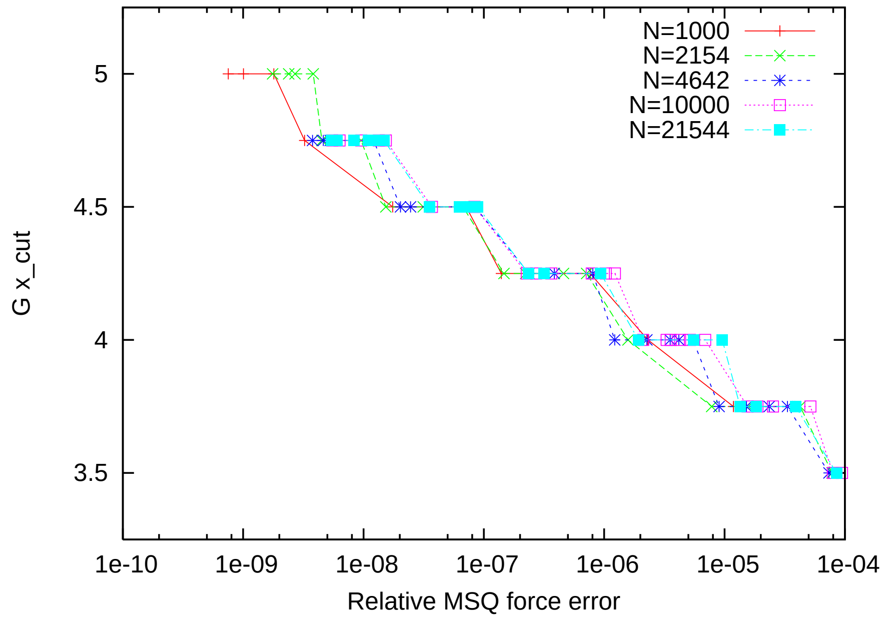

Figure 4: The optimal values of Gx_{cut}, crystal system.

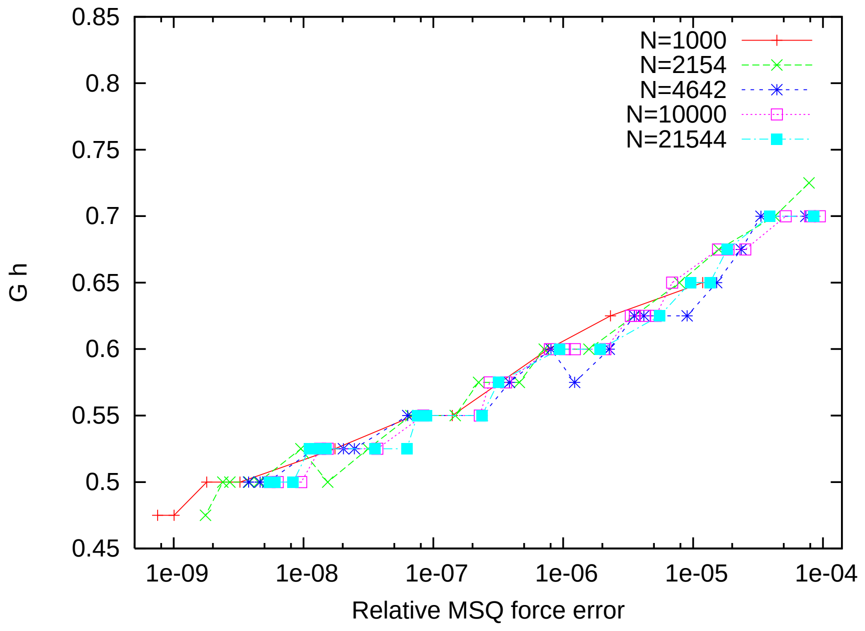

Figure 5: The optimal values of Gh, crystal system.

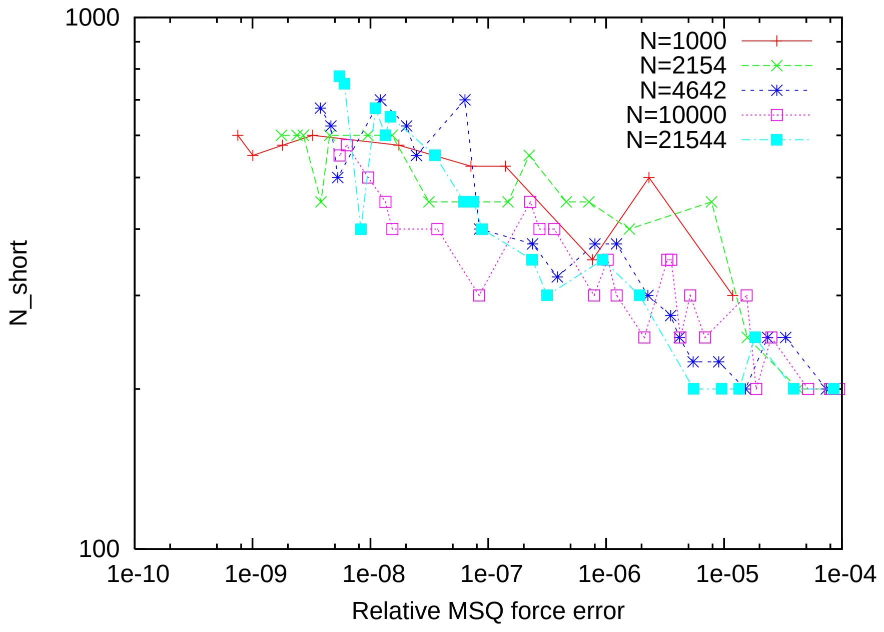

Figure 6: The optimal values of N_{short}, crystal system.

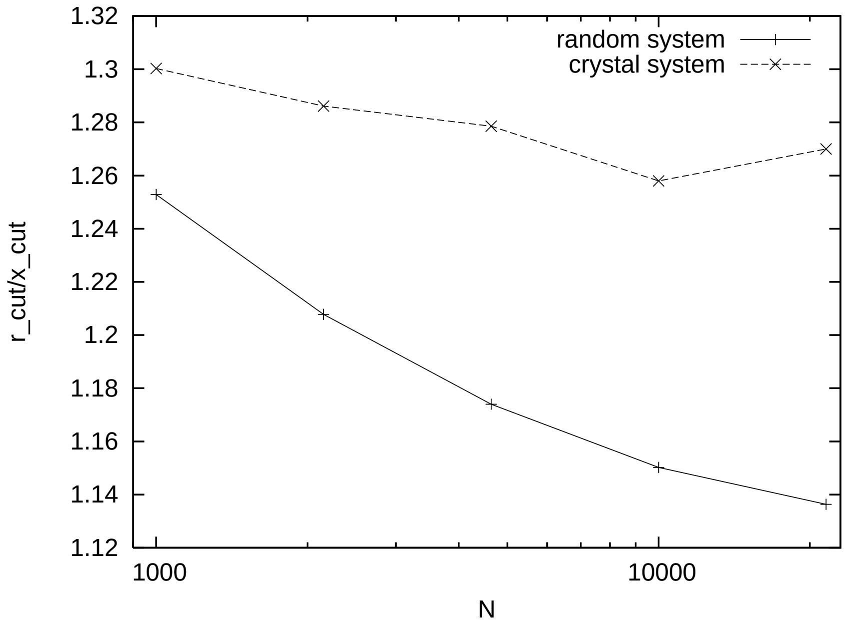

Figure 7: Values of $r_{\rm cut}/x_{\rm cut}$ for both random and crystal systems.

This is similar to finding the optimal value of the Gaussian width in the standard particle-mesh schemes [24]. Finally, the optimal ratio $r_{\rm cut}/x_{\rm cut}$ was observed to be independent on the required accuracy but slightly decreased with rising N, as seen on Fig. 7.

The Pareto frontiers allow us to determine the crossover points for each N: the values of MSQ force error for which our calculation takes the same time as the full direct one. They are presented at Fig.8. We have plotted the crossover curves for the random system and for the crystal system. For comparison we have also plotted the crossover curve of the fast multipole method taken from [5], where one of the best implementations of FMM is described. In that paper the same charge distribution was used as in our random system. Of course, the random positions of the particles where different in our tests and in those of [5]. However, we still see that our method has lower crossovers than the FMM for the same accuracy.

9 The energy conservation.

In contrast to other methods such as FMM [1], [34], our algorithm has the advantage that the approximate forces are exact analytic derivatives of the approximate energy. This allows for energy conservation during an MD run. To illustrate this, we make an MD simulation of a rock salt crystal formed by 1000 Na and Cl atoms. The particle positions and velocities are updated by the velocity Verlet algorithm.

To get physically reasonable results, we made the particles interact through the Born-Mayer-Huggins-Fumi-Tosi (BMHFT) rigid-ion potential [58] that has

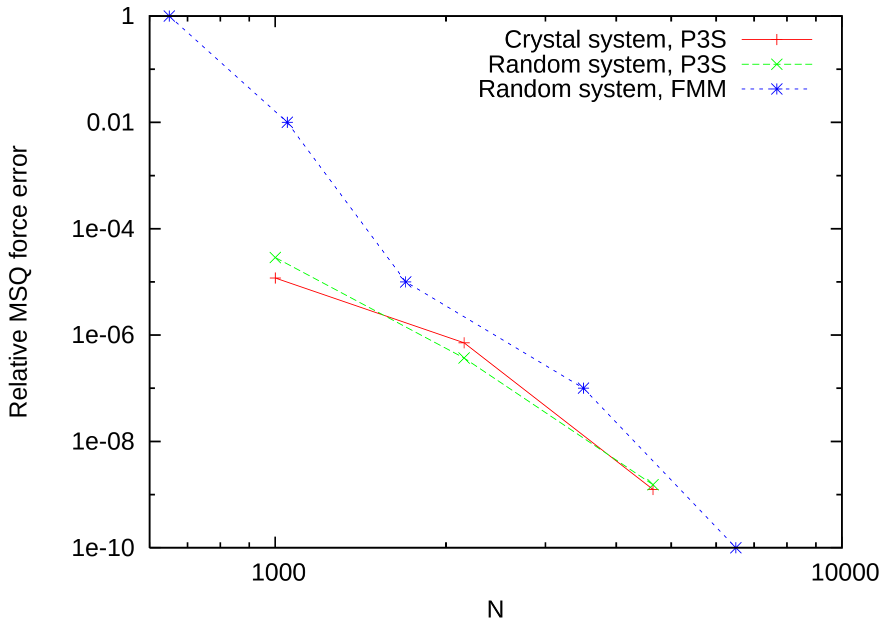

Figure 8: The crossover curves.

bonding terms in addition to the Coulombic force.

At first we made the system equilibrate for 300 oscillation periods. We then monitored the potential and total energy for another 100 periods using the full direct algorithm. Then the last 100 periods were repeated using our P3S algorithm.

On Fig. 9, 10 we plot the absolute values of deviations of the potential and total energy from their mean values. The ratio of the mean square deviation of the total energy to that of the potential one is found to be equal to 1.4×10^{-3}.

The P3S results are shown on the graphs; those of the full direct calculation are indistinguishable from the P3S ones.

10 Parallelization.

The parallelization of the calculation of the short-range energy and forces is straightforward: for the particle i and processor IPROC, we calculate the force only if $\mathrm{MODULO(I, NPROC) = IPROC}$, where NPROC is the total number of processors. In the end, an MPI_ALLREDUCE command sums all the contributions.

For the long-range part, we rely on the parallel structure of the Poisson solver [51]. The charge density on the grid is divided into slabs in the $x - y$ plane. Each such slab is the input density at a separate node. The Poisson equation (for the whole cell) is solved for that slab density. The output at a given node again contains only the corresponding slab of the grid, this time it is a piece of the potential array. Of course one needs global interprocessor communications, including MPI_ALLTOALL for the solution of the Poisson

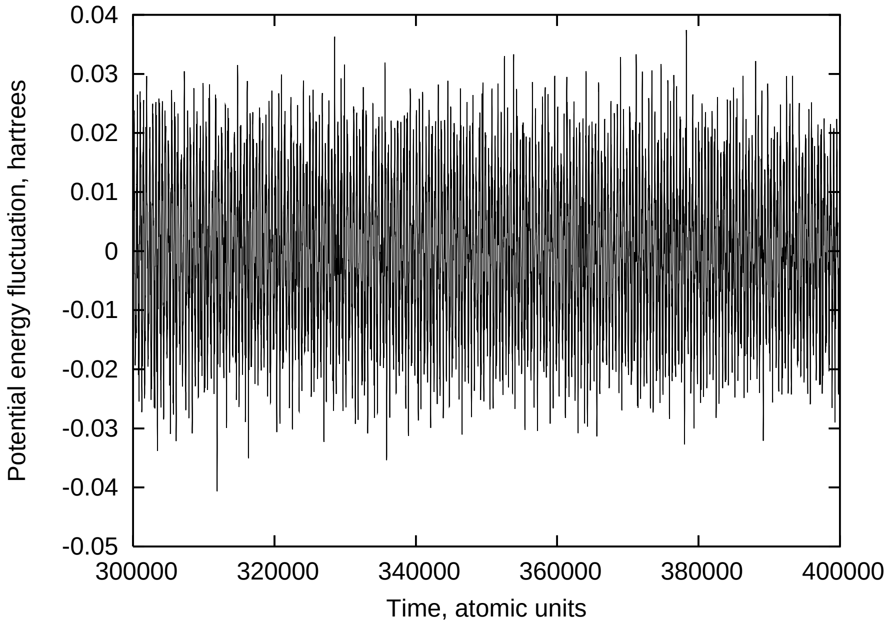

Figure 9: The potential energy fluctuations with the P3S method.

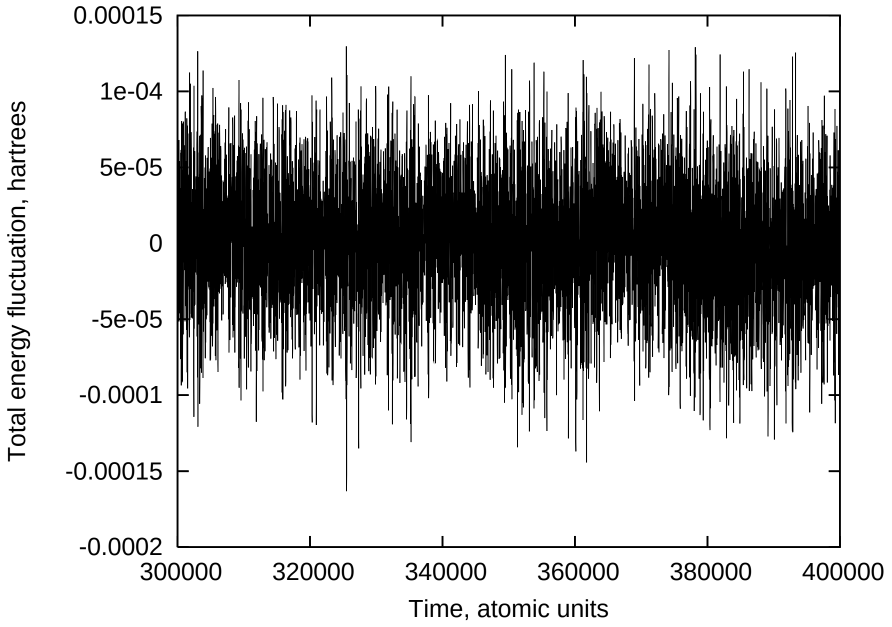

Figure 10: The total energy fluctuations with the P3S method.

equation.

Then in the original algorithm of [51], an MPI_ALLREDUCE command sums up the potentials of all the slabs.

In our program for point particles, we kept the parallelization of the charge assignment as above: each processor receives only the grid charges from the corresponding slab. However, we do not use the MPI_ALLREDUCE command to get the potential. Instead, we used the slabwise structure of the output of the Poisson solver. For each node, the corresponding slab potential contributes only to the forces on particles that are close to it.

In the end, we add up the *forces* with the MPI_ALLREDUCE command. This MPI_ALLREDUCE is actually merged in the program with the one for the short-range forces.

In this way we minimize the interprocessor communication considerably compared e.g. to [22] since it is much easier to send the components of the forces than the pieces of the enormous potential array.

To test the parallel program we ran it for the random system (the same as in the serial tests) with 10^5 particles. The relative accuracy of the forces was kept around 10^{-6}.

The result is given at Fig.11. At the y axis we have the ratio of the CPU time spent on several processors to that spent by one processor.

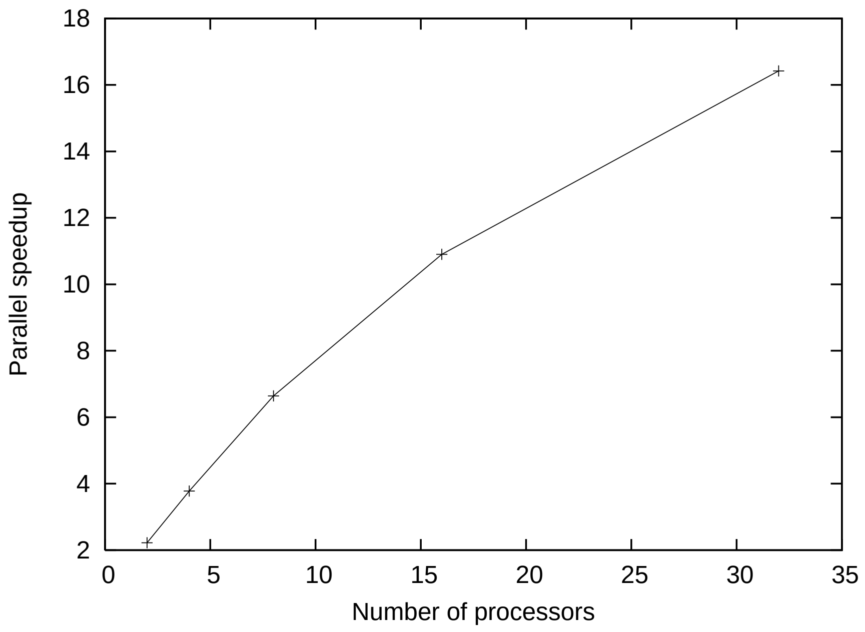

Figure 11: The parallel speedup results on a CRAY XT3 for 10^5 particles.

11 Conclusion.

We have developed a point particle Poisson solver algorithm that has a lower crossover point than the FMM. It can be considered as a generalization of the particle-mesh solvers for free boundary conditions. It can also achieve high precision.

The forces obtained by our program are analytical derivatives of the energy; this is an advantage in the context of MD simulations. An MPI parallelised version of the algorithm is presented that scales well on a moderate number of processors.

12 Acknowledgments.

The authors acknowledge financial support from the Swiss National Science Foundation. The numerical calculations were done on a parallel supercomputer of the Swiss National Supercomputing Center.

References

[1] C. Sagui and T.A. Darden, Annu. Rev. Biophys. Biomol. Struct. **28**, 155 (1999).

[2] R. Spurzem and A. Kugel, Astronomisches Rechnen-Institut Heidelberg, preprint No. 95, 1999.

[3] L. Greengard and V. Rokhlin, J. Comput. Phys. **73**, 325-348 (1987).

[4] H.-Q. Ding, N. Karasawa, and W. Goddard, J. Chem Phys, **97**, 4309-4315, 1992.

[5] C. A. White and M. Head-Gordon, J. Chem. Phys. **101**, 6593 (1994).

[6] J. Shimada, H. Kaneko, and T. Takada, J. Comp. Chem. **15**, 28 (1994).

[7] Greengard LF, Science **265**, 90914 (1994).

[8] S. Pfalzner and P. Gibbon, *Many-body tree methods in physics* (Cambridge University Press, 1996).

[9] K. Esselink, Comp. Phys. Commun. **87**, 375 (1995).

[10] J. Barnes and P. Hut, Nature **324**, 446 (1986).

[11] P.Gibbon and G. Sutmann, in *Quantum Simulations of Complex Many-Body Systems: From Theory to Algorithms*, J. Grotendorst, D. Marx, A. Muramatsu (eds.) John von Neumann Institute for Computing, Jülich, NIC Series, **10**, ISBN 3-00-009057-6, pp. 467-506, 2002.

[12] D. Wolf, P. Keblinski, S. R. Phillpot, and J. Eggelbrecht, J. Chem. Phys. **110**, 8255 (1999).

[13] C.J. Fennell and J.D. Gezelter, J. Chem. Phys. **124**, 234104 (2006)

[14] P.P. Ewald, Ann. Phys. (Leipzig) **64**, 253 (1921).

[15] M. P. Allen and D. J. Tildesley, *Computer simulation of liquids* (Clarendon Press, New York, 1988).

[16] A. Toukmaji and J.A. Board, Ewald sum techniques in perspective: a survey, Comp. Phys. Comm. **95**, 78 (1996)

[17] R. W. Hockney and J. W. Eastwood, *Computer Simulation Using Particles* (IOP, Bristol, 1988).

[18] T.A. Darden, D. M. York, and L. G. Pedersen, J. Chem. Phys. **98**, 10089 (1993).

[19] D. York and W.Yang , J. Chem. Phys. **101**, 3298 (1994)

[20] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. Pedersen, J. Chem. Phys. 103, **8577** (1995).

[21] P. MacNeice, *Particle-Mesh Techniques*, NASA Contractor Report 4666 (1995).

[22] E. L. Pollock and J. N. Glosli, Comp. Phys. Comm. **95**, 93 (1996)

[23] M. Deserno and C. Holm, J. Chem. Phys. **109**, 7678 (1998).

[24] P.H. Hünenberger, J.Chem.Phys. **113**, 10464 (2000).

[25] C. Sagui and T. Darden, J. Chem. Phys., **114**, 6578 (2001).

[26] R. D. Skeel, I. Tezcan, and D. J. Hardy, J. Comp. Chem. **23**, 673 (2002).

[27] S. Banerjee and J. A. Board, J. Comput. Chem. **26**, 957 (2005).

[28] Y. Shan, J.L. Klepeis, M.P. Eastwood, R.O. Dror, and D.E. Shaw, J. Chem. Phys. **122**, 054101 (2005).

[29] Z.-H. Duan and R. Krasny J. Chem. Phys. **113**, 3492 (2000).

[30] F. Hedman and A. Laaksonen, Chem. Phys. Lett. **425**, 142 (2006).

[31] K. E. Schmidt and M. A. Lee, J. Stat. Phys. **63**, 1223 (1991).

[32] F. Figuerido, R. M. Levy, R. Zhou, and B. Berne, J. Chem. Phys. **106**, 9835 (1997).

[33] M. Challacombe, C. White, and M. Head-Gordon, J. Chem. Phys. **107**, **10131** (1997).

[34] T. C. Bishop, R. D. Skeel, and K. Schulten, J. Comp. Chem. **18**, 1785 (1997).

[35] A. Toukmaji, D. Paul, and J. A. Board Jr., Duke University Technical Report 96-002.

[36] I.F. Sbalzarini, J.H. Walther, M. Bergdorf, S.E. Hieber, E.M. Kotsalis, and P. Koumoutsakos, J. Comp. Phys. **215**, 566 (2006).

[37] A.V. Philippov, V.V. Vatutin, and R.A. Veselov, Rev. Sci. Instr. **75**, 1499 (2004).

[38] J.A. Board, J. W. Causey, J.F. Leathrum, A. Windemuth, and K. Schulten, Chem. Phys. Lett. **198**, 89 (1992)

[39] H. Schwichtenberg, G. Winter, and H. Wallmeier, Parall. Comp. **25**, 535 (1999).

[40] J.A. Lupo, Z. Wang, A.M. McKenney, R. Pachter, and W. Mattson, J. Molec. Graph. and Model. **21**, 89 (2002).

[41] S. Ogata, T. J. Campbell, R. K. Kalia, A. Nakano, P. Vashishta, and S. Vemparala, Comp. Phys. Commun. **153**, 445 (2003).

[42] J. Kurzak, B. M. Pettitt, J. Comp. Phys., **203**, 731 (2005).

[43] J. Kurzak, B. M. Pettitt, J. Parallel Distrib. Comput. **65**, 870 (2005).

[44] P.H. Hünenberger and J.A. McCammon, J. Chem. Phys. **110**, 1856 (1999).

[45] P.H. Hünenberger and J.A. McCammon, Biophys. Chem. **78**, 69-88 (1999).

[46] R.W. Hockney, Methods Comput. Phys. **9**, 135 (1970)

[47] G. J. Martyna and M. E. Tuckerman, J. Chem. Phys. **110**, 2810 (1999).

[48] G. Sutmann and B. Steffen , Comp. Phys. Comm. **169**, 343 (2005)

[49] J. Beckers, C. Loewe, and S. de Leeuw, Mol. Simul. **6**, 369 (1998).

[50] U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid* (Academic Press, San Diego, 2001).

[51] L. Genovese, T. Deutsch, A. Neelov, S. Goedecker, and G. Beylkin, J. Chem. Phys. **125**, 074105 (2006).

[52] T.N. Heinz and P.H. Hünenberger, J. Comput. Chem. **25**, 1474 (2004).

[53] Z. Yao, J.-S. Wang, G.-R. Liu, and M. Cheng, Comp. Phys. Commun. **161**, 27 (2004).

[54] G. Deslauriers and S. Dubuc, Constr. Approx. **5**, 49 (1989).

[55] S. Goedecker, *Wavelets and Their Application* (Presses polytechniques et universitaires romandes, Lausanne, 1998).

[56] J. Kolafa and J. W. Perram, Mol. Simul. **9**, 351 (1992).

[57] H. G. Petersen, J. Chem. Phys. **103**, 3668 (1995).

[58] F. G. Fumi and M. P. Tosi, J. Phys. Chem. Solids **25**, 45 (1964).